计算机硬件基础

（下）

计算机应用软件人员水平考试应试指导
联合会教材编写组

上海微电脑厂

第六章　　控制器

数字计算机最显著的特点，就在于它是一种高度自动化的工具，能够自动地，连续地进行工作。数字计算机具有这种特点主要是由于它采用了“程序控制”结构的缘故。

所谓“程序控制”，是指计算机通过一个复杂的自动控制系统，依据程序安排好的指令序列的执行顺序，逐条地、自动地、连续地进行计算，直到获得全部运算结果并输出为止。这个复杂的自动控制系统的主要部分就是计算机的控制器。

控制器做为全机的控制中心，它根据指令的要求指挥全机工作。在工作过程中出现的意外事件，如出错，掉电等，控制器也应能做适当的反应。

控制器的具体结构取决于计算机的指令系统，总体布置、工作节拍，控制方式和微操作信号产生方式等诸多因素，所以不同机器的控制器结构上存在很大差别。本章主要讨论组成控制器的一般原理。

一、指令分析

我们知道，指令是以二进制代码的形式存放在存贮器中的，因此，在执行一条命令之前，必须把这条指令从存贮器取到控制器，把它放在指令寄存器里。这时，代表操作性质（例如+、－、×、÷、…）的操作码，仍然是指令寄存器前几位所存放的一个二进制代码。指令分析的任务就在于把这组代码，变成它所对应操作的控制电位。这是一个典型的多——译码问题，所以分析操作性质的任务是由操作译码器来完成的。

例如操作码为六位二进制代码，采用八进制写法，共编码从“00”到“77”共64种；因此最多对应64种不同的指令。假如加法指令的操作码为“01”，当加法指令取到控制器以后，通过操作译码器进行译码，则唯一地在加法控制线上出现高电平，而其它控

制线均为低电平。这样来控制机器进行加法运算。因此，指令分析的过程，就是计算机识别操作性质的过程，而这一过程是通过操作译码器来实现的。

二、地址形成

计算机在执行指令的过程中，需要不断地向存贮器取指令、取数和存数，而每取一条指令或存取一个数码，都必须向存贮器发送它所在的存贮单元的地址。从控制器的角度来看，地址可以分为两种，一种是指令地址，另一种是数码的地址。

指令地址的形成是很简单的，因为机器是按顺序逐条执行指令的，在一般情况下执行顺序就是指令在存贮器内存放的地址顺序。所以，通常我们采用计数器来存放指令地址，称为指令计数器。开始时，将一条指令地址送入；以后每执行完一条指令，将指令计数器加“1”则形成要执行的下一条指令所在地址。如果计算中需要改变这个顺序时，只要通过转移指令送入一个新的地址（称为转移地址）即可实现。

数码地址的形成则是比较复杂的，总的来说，它是在指令变址特征位的控制之下，通过地址寄存器、变址器和变址加法器（统称控制器的变址部分）来实现的，它是控制器的一个重要组成部分。

三、脉冲产生

电子计算机的计算过程，从宏观（程序控制）来看是指令序列的执行过程；而从微观（指令控制）来看，一条指令总是由若干简单操作（称为微操作）所合成的，因此又是一个微操作序列的执行过程。计算机的基本控制方法，总是将执行一条指令的总时间划分为若干个时间区间，而在每一时间区间产生一个“节拍电位”和一个“时钟脉冲”，用来控制产生微操作。上述任务是由控制器的脉冲分配器和脉冲发生器来完成的。因此，脉冲分配器可以看成是整个机器的“心脏”，随着它的“跳动”整个机器进行有条不紊的工作。

四、微操作控制

不同的指令是由不同的微操作序列所组成的，不同的指令的执行过程就是不同的微操作序列的执行过程。对于一台计算机，往往具有几十条甚至上百条指令，在设计它的控制部分时，其主要的工作量就在于逐条地排出它的微操作序列。每一个微操作都是操作译码器译出来的操作电位 Zi ，和脉冲分配器产生的节拍电位 Wi ，时钟脉冲 mi 以及控制条件的组合，要求做到正确，合理、节约。这个部分称为控制器的微操作控制部分，它是由大量起控制作用的门电路所组成的。

五、人机联系

电子计算机的控制系统，除了自动控制部分以外，还需要人工控制部分。这个部分就是控制台及其它一些附属设备（如打字机等）。操作者可以通过这个部分在计算机前将程序送入机器，还可以通过这个部分监视运算过程，以及在运算过程中发现问题对程序进行必要的修改和处理。另一方面，我们对一台生产出来的计算机进行调整，以及交付使用以后的维护工作，都需要借助于这个部分，所以它是人和机器进行联系的一个重要渠道。

6·2 指令及指令系统

CPU 的主要任务是执行指令，而实施指令执行的则是控制器的功能，控制器根据指令译码产生一段微程序序列，以完成指令所指定的操作。显然，指令的结构形式，繁简程度，指令的条数和规模等因素都与控制器的构成密切相关，下面，我们将对指令的形式及指令系统作一较全面的介绍，因为它是计算机设计中的根本性问题，是计算机软硬件的一个界面。

一、指令的形式

指令是命令计算机进行某一具体操作（如十、一，×，÷，取数，存数）的命令，因此它是计算机程序部最基本的单元。指令序列中的

每一条指令，都是以二进制代码形式存贮在内存中的，执行时将单元中的指令取出送到指令寄存器中寄存起来，然后由控制器“译码”产生一段相应的微程序序列。那么计算机的指令应具什么形式，一条指令应该包括哪几个部分呢？一般来说，指令应有两个部分组成，第一部分要指的指令执行的操作，即操作码部分，另一部分，要指的操作的数据对象，即地址码部分，两者都用代码表示，具体形式如下：

| 操作码 | 地址码 |

我们一般习惯于用 ϑ 表示操作码，用 D 表示地址码。由于每条指令进行一种操作，所以每条指令中的操作码 ϑ 是唯一的，但是不同的计算机，不同的指令，地址码部分所表示的地址的个数是不同的。所以计算机按照地址的个数分类，就可有三地址机器，二地址机器和一地址机器等不同类型，现在我们简单地介绍这几种不同类型的指令。

1．三地址指令

这类指令的地址码部分包括三个地址，分别称为第一地址（D_1）第二地址（D_2）和第三地址（D_3），又称为第一操作数地址，第二操作数地址和结果地址，其形成如下：

| ϑ | D_1 | D_2 | D_3 |

其中 ϑ 表示一种操作，这条指令所执行的功能是：

$$(D_1)\,\vartheta\,(D_2) \longrightarrow D_3$$

表示将内存单元 D_1 存放的数和内存单元 D_2 存放的数取得运算器两个输入端口，然后进行 ϑ 性质的运算，结果将送回到内存单元 D_3 中去。这样的运算方法比较直观，使用也比较方便，但是指令长度较长，一条指令占据内存单元多，执行时访问内存次数也较多，执行指令的速度要受到影响。但是另一方面，並不是所有的地址都需要

有三个地址，有些指令可能只要操作码就可以了。

2. 二地址指令

所谓二地址指令，那地址部分只要二个地址，其形式如下：

| ϑ | D_1 | D_2 |

这类指令功能和三地址指令功能是类似的，D_2 单元还用作结果地址，即运算结果存入 D_2，直观的指令功能如下：

$$(D_1)\ \vartheta\ (D_2) \longrightarrow D_2$$

D_1 通常称为源操作数地址，D_2 称为目的操作数地址。二地址指令同样是较直观的，使用上也还方便的。但是同样存在指令长度较长，访问器存数次多（这样指令执行时要访问器存三次）的问题。

3. 一地址指令

一地址指令的形式如下：

| ϑ | D |

通常，以算术运算为例，操作需二个数据，而这类指令中只提供一个数据地址，而另一个地址是指令系统中隐含指定的，总是上次运算的结果保留在运算器的累加器 L 中），二者运算后的结果仍保留在累加器 L 中。其功能如下：

$$(L)\ \vartheta\ (D) \longrightarrow L$$

这类指令的形式是不直观的，相对于多地址指令来说是有一定的不便之处，例如要执行这样的运算

$$(D_3) = (D_1) + (D_2)$$

用三地址指令只要一条指令就可以完成。

| ADD | D_1 | D_2 | D_3 |

假定 ADD 是加法的操作码。

而如果用一地址指令来完成这个运算，则需三条指令，即：

$(D_1) \longrightarrow L$；将 (D_1) 内容取到 L

$(L) + (D_2) \longrightarrow L$；做 $(L) + (D_2) \longrightarrow L$

$(L) \longrightarrow D_3$；将和送到 D_3。

但是实践事例证明，大量的运算是和上次运算结果进行累加运算的，这样，上述的矛盾对整个问题过程的影响并不很突出，即使要执行上述运算，其数量通常是不很多的。

一、地址机器的主要优点可综合如下：

I）控制简单。多数指令统一访问内存两次，次数少而且控制简单。

II）结构合理。由于只有一个地址，所以不同类型的指令的地址部分得到有效的利用。同时指令长度外，占据内存单元少，从而使内存得到充分的合理的利用。

III）运算速度快。由于一地址指令访问内存单元次数少，所以在采用同样开关速度元件的条件下，一地址机器就较二地址、三地址机器的平均运算速度要高。

由于一地址机器在实际运算中利用率较高，所以有许多计算机采用一地址指令。以后的有关于指令的讨论中，大都是针对一地址指令而言的。但是任何都具有相反的二个方面，虽然一地址机器具有以上一些优点，但是在灵活性和使用的方便性以及形式的直观性等方面，一地址机器还存在着一定的缺点。为介决这矛盾，通常采用一种折中的办法，所以 出现了其他地址类型的机器，这里不再一一讲述了。

二、指令系统

一台计算机所有能执行的指令，称为这台计算机的指令集，或算这台计算机的指令系统。不同的计算机通常具有不同的指令系统。一台计算机的指令系统中所含指令的条数，可以通过操作码占二进位的位数反映出来。若操作码占5位二进制位，则该机的指令系统最多

能包含了2条指令。若操作码占7位，则最多包含$2^7=128$条指令。

通常来说，指令条数的多少反映出计算机实现功能的强弱。但是，随着指令数目的增多，性能扩充，则必须会引起控制系统复杂，设备增多。一般计算机的指令系统可根据指令的功能进行划分成几个部分，它们是数据代码传送类指令，算术及逻辑运算类指令，转移控制类指令，外部设备命令及其他类型指令。下面简单介绍有关类型的指令。

1．数据代码传送类指令

传送类指令可以说是计算中最基本的指令，它的结构可能具有下列形式：

传送	地址部分	字级指示	入—出方式

上述地址部分是规定了一个传送的存贮器地址，地址的具体数值可由方式码选择来得到。

字段指示部分是指明传送是按照位、字节、半字、字、双字、四字进行传送的。若是数据字块传送，还应指出字块的长度。在个别情况下，还可以指出传送的是浮点数的阶码或尾数。

数据传送的“源”和“目的”可以是任一存贮单元或某寄存器，这决定于计算机的地址结构以及其隐或显表示地址的方式。

如果“源”字段和“目的”字段不等长，则指令中必须指出不等长的调整方式。例如把两个字送入一个四字长的区域中，是从左边（高位）取齐还是从右（低位）取齐，留出的空位如何处理，是填入∅还是填入1或是其他值等等。凡是不等长字段间的各种操作，都需要指出调整方式。对于面向字节的计算机，调整时常是一项非常重要的功能。

下面我们给出一张表格来说明指令系统中最基本的指令的名称，符号、和功能。在给出这张表格之前，关于内存地址单元编号进行说明：通常内存编号是以字节为单位的，而字单元可以是以二字节，三或四字节组成，这里假定以二字节组成一个字单元，则一个字单元就包含有二个地址编号，左半单元地址为K，右半单元地址为K＋1。

由于地址编号通常从“力”开始，所以K总为偶数，而K＋1则总是奇数。传送是每次以一个单元为单位的，因此要存取二个地址号的内容。

符　号	指令名称	操作内容	说　　明
→L	取　数	(D,D+1)→L	内存D，D+1单元所存代码送运算器的累加寄存器L
→L_Z	半字长取数	(D)→L_Z	内存D单元所存代码送L左半部分
L→	存　数	(L)→D,D+1	运算器的累加寄存器L所有代码送内存D，D+1单元
L_Z→	半字长存数	(L_Z)→D	L所有代码左半部分送内存D单元
0→	存存　零	0→D，D＋1	将内存D，D＋1单元清除为零
B→L_Z	变址器送L	(b)→L_Z	变址器b单元的所存代码送L寄存器左半部分
L_Z→B	(L)送变址器	(L_Z)→b	L所存代码左半部分送变址器b单元
d→B	d送变址器	d→b	指令的地址码d送变址器b单元
L→S	(L)送S	(L)→S	L寄存器所存代码送S寄存器
S→L	(S)送L	(S)→L	S寄存器所存代码送L寄存器

续

+	浮点规舍加	(L)+(D,D+1)→L	寄存器L中代码加上内存D，D＋1单元所存代码，和数存入L
−	浮点规舍减	(L)−(D,D+1)→L	(L)减去(D，D＋1)，差存入L
×	浮点规舍乘	(L)×(D,D+1)→L	(L)乘以(D，D＋1)，乘积存入L
/	浮点规舍除	(L)/(D,D+1)→L	(L)除以(D，D＋1)，商存入L
	浮点规舍反减	(D,D+1)−(L)→L	(D，D＋1)减去(L)，差存入L
M−	浮点规舍模减	|(L)|−|(D,D+1)|→L	(L)的绝对值减去(D，D＋1)的绝对值，差差存入L
L^2	浮点规舍自乘	(L)×(L)→L	(L)的平方送入L
	浮点规舍反除	(D,D+1)/(L)→L	(D，D＋1)除以(L)，商存入L
T_+	整数加	$(L_Z)\ T_+(D)\rightarrow L$	(L)的左半部分加上(D)，和数存入L
T_-	整数减	$(L_Z)\ T_-(D)\rightarrow L$	(L)的左半部分减去(D)，差存入L

2. 算术及逻辑运算类指令

算术运算包括基本的算术运算以及和运算有关的各种操作，算术运算包括＋、－、×、÷等等运算。其一般结构如下：

运算指示部分表示进行何种算术运算。

方式可以是定点，浮点；数的表示法可以是二进制、十进制。方式中还可包括半字长运算，全字长运算、双字长（精密）和 字长（超精密）运算等。

调整部分如同前所述。

算术运算可能有遗留问题、如除法时有余数，乘法时可能数值过长，浮点对阶时尾数可能小于最小有效值等。对于这些情况必须有处理的约定，是舍去还是保留于指定的地方？舍去后，保留的数如何处理等等，这些约定有时要在余数部分中指出。

逻辑运算又称为二元布尔代数运算，其运算对象只能取二种可能值，或0或1。并且布尔运算是封闭的。逻辑运算共有16种，但是只有极少机器齐备这16种运算。在这16种逻辑运算中主要的操作是：置1，置0、与、或、异或、等价和求外几种。

3，转移控制类指令

控制器是按照指令在内存的顺序执行的，除非程序要求打断这种顺序性。若要打断这种顺序执行结构，则必须转移指令。转移指令的目的就是要跳过一段指令，而去执行规定地址中的指令。转移指令可分为条件转移和无条件转移二种。

条件转移的判断条件通常有：＞、≧、＝、≦、＜、上溢、下溢、奇偶差，奇、偶，正数、负数、整数、小数、进位、和各种逻辑比较、某些触发器的状态以及任何需要进行判断的其他情况。无条件转移是条件转移的一种特例，其条件码是空条件。转移指令通常可实现分支、循环等运算。

控制指令又称程序控制指令，其用于控制程序的顺序，或用于变更对后面的指令的介释以及其他类似的用途。

控制指令主要有以下几种：

方式控制指令：它类似于键盘中的换挡键，换挡键的作用是同一

字符键可以获得不同的字符输出。如果一个计算机的字长很短，安排不了所需要的各种形式的指令，这样就可安排一条方式控制指令来增加指令的花式。在方式控制指令后面 出现的指令，要按照出现以前的不同方式来介释。例如对于算术运算指令，可由二进制变为十进制或由定点变为浮点等等。

子程序的链接指令，或称转子指令，子程序的概念就是一段程序共用指令，其可使程序结构清晰，代码简清优化等。转子指令同一般的转移指令的区别在于：转子指令要为返回主程序做出必要的安排，它的关键问题在于保存返回主程序的地址，另外，还需考虑其他一系列保护措施。

4. 外部设备命令

外部设备是指与计算机相连的外部世界，外部设备是指输入、输出、外存等部分，输入与输出是计算机系统的瓶胫，它们担负着大量代码的输入，输出及其转贮信息等任务。因此 I/O 指令在近代计算机系统中占有相当大的比例。

I/O 的指令一般包括：

I）数据输入，把 I/O 数据读入到存贮器的指令，数据可以存于几个 I/O 设备中，用这种指令可以将数据收集起来，并存贮在存贮器指定的地点。

II）数据输入。将存贮数据送到 I/O 设备。I/O 完成的动作可以是写入、打印、显示，凿孔或其他，随着设备的不同而异。

III）操作指令。命令 I/O 设备进行某种 动作而不进行 数据传送，如卷带、快速前进、清除显示、脱机等等。

5. 其他类型的指令

这类指令如空操作（KOn）、停机指令（T），符合停机指令（TT）等等。停机和符合停机指令又称为控制台指令，方便程序员的操作和调试检查程序的执行情况。

6·3 地址的形成

大家知道在程序的执行过程中，需要不断地从内存中取出指令和取数、存数，因此就要求不断地将指令地址和操作数地址送往内存的地址寄存器。控制器的一个重要功能就是不断地形成指令地址和操作数地址，并送给内存的地址寄贮器，其中特别是数码地址的形成是比较复杂的。下面讨论上述二个地址的形成问题。

一、操作码地址又称指令地址。我们知道计算机的计算过程，是按照事先编好并放在内存里的程序的顺序，逐条执行指令的过程，每执行完一条指令，都需形成下一条指令的地址，以便使程序自动连续地运行。而下一条指令的地址总是现行指令地址加“1”，除非是一条转移指令。控制器中的“指令计数器”（IC）担任了自动形成指令地址的任务，其内容就是下条指令的地址。每取出一条指令后，指令计数器就自动加“1”，以形成下一条指令的地址。

在转移指令的情况下，则需要简单地做一次传送：将转移地址D送到指令计数器中去，以形成转移地址，这样就完成了转移的功能。

二、返回地址的形成

程序转到子程序去执行时，需要保存返回地址，以实现子程序的自动返回，实现“自动返回”的指令，具体来说有二种，一种是使用“后进先出”缓存区的转子指令和返回指令；另一种是不使用“后进先出”缓存区的转返指令。目前大多数机器使用的是前一种，下面我们对前一种进行讨论如下。

1．“LIFO”存贮器

“LIFO”是一种存贮器的特殊工作方式，其与随机存贮器不同，总可以预测到下一次传送的“LIFO”存贮器的地址。这个地址通常称为堆地址，存贮在某一特定寄存器中。“LIFO”存贮器的工作方式综合起来有以下几点：

I）存数按地址顺序存放。

II）取数按相反顺序取出。

Ⅲ）存取地址由堆指针指示，遇“读令”则减“1”形成操作地址，进行取数。遇“写令”则在存入代码后加“1”形成新的堆地址。

2。“LIFO”存贮器的作用

“LIFO”区的一个重要用途就是存放返回地址，实现程序的自动返回、特别是在多重嵌套的情况下，使用它进行这种管理是很方便的。另外，“LIFO”作用存贮器的系统部分，由系统统一管理，因此又是保护数据的重要场所，不致由于考虑不周而引起程序数据的破坏。

三、数码地址的形成

1. 指令中的方式码

近代计算机指令系统的一个显著特点是主要地不用增加指令类型的方法来扩大指令系统的功能，而是利用各种形式化编码（Modality）或叫方式码来使一条指令的功能多样化，从而适用于多种用途。其中最突出的是寻址方式的选择。

2. 寻址方式

所谓寻址方式是指形成指令操作数码地址的方式。使用寻址方式，可以在较短的地址部分访问较大的存贮区域。若不使用寻址方式而采用直接地址，则64 K存贮就需16位地址码，不利于指令的缩短，从而引起其他有关的一条列影响。如果采用寻址方式，假定使用变址寻址，则只需要几位用于选择一个变址器，而变址器却可以访问到很大的存贮区域。

寻址方式主要有：变址寻址方式、间址寻址方式、立即地址等等。这些寻址方式并不是互斥的，有些可以在一条指令中共存而不会引起形成地址的困难。

Ⅰ）变址寻址方式

近代计算机多数设置有变址寄存器，将变址寄存器的内容与指令中的形式地址部分进行指定的运算而得到实际操作的内存地址，称为有效地址。对于一般计算机，这种操作仅限于求和与求差。但是变址

做为一个重要概念它并不局限于简单的求和与求差，可能包括有许多其他较复杂的运算。

另一种变址方式是将变址寄贮器的内容作为有效地址的高位，而形式地址为低位，二者高低合并联合形成一个有效地址。这种变址方式的逻辑上对应是存贮器分页概念。因此又称页面寻址。变址寄贮器的内容是需要进行读写单元所在的页面地址，而形式地址是该单元相对于该页面的页内地址，因此，变址寄存器又称为页址寄存器。例如页址寄存器内容为 078，

00，111，101

而形式地址为 164。

01，110，100

则实际有效地址为 036564。

0，011，110，101，110，100

Ⅱ）间址寻址方式

所谓间址是指给出的地址并不是实际的有效地址，而是操作数地址的地址，因此是一个地址转接。间址寻址也是一种重要的寻址方式。是否间接地址用指令字中某指定位是 1 或 0 来表示，1 表示间接地址，0 表示有效地址，这样可实现地址的多重间接。间址和变址经常是共存的。典型的变址操作应在间址之前进行，如下列取数指令 LD

$$(L)=(((IRD)+(X)))$$

其中 L 表示累加器、X 表示变址寄存器，IRD 为地址寄存器，每一对括号表示一次间址。

Ⅲ）立即操作数

一般来说、指令的地址部分给出的是一个操作数地址，而不是操作数本身，但是也不排除相反的情况，即地址部分直接给出了操作数

本身，而不是它的地址。这就是立即数寻址方式。

另外还有许多其他的寻址方式，例如，寄存器寻址，直接寻址和相关寻址等寻址方式，这里不再逐一介绍。

6·4 指令译码及控制

计算机在运算过程中，从内存每取出一条指令，都送到控制器的指令寄存器 J_Z，控制器首先要对这条指令进行分析，第一根据操作码，分析指令的操作性质，通常称为操作译码。第二根据变址特征位形成操作数地址并发送到地址寄存器 J_{DN} 中，以便存取数码。

一、操作译码

前面已经说明过。计算机的指令是用二进制表示的，并且指令的操作码部分是一一对应的，如空操作为 $\varnothing\varnothing_8$，十法为 $\varnothing 1_8$、停机 T 为 7_8。为了识别指令操作的性质，可以通过指令寄存器中的指令进行译码，从而将一组操作代码转换成一个特定的控制电位。如图 6－1 所示。操作码共有六位，可表示 64 种不同的操作，采生 64 种不同的控制电位。译码选中的电位线为高电平，其平均为低电平。

二、读写控制

每当存取内存某一单元时，控制器需要发出一些控制信号。这些控制信号所完成的动作顺序是：

1. 发送内存地址

每存取一个代码，无论是取指令，还是存取数据，都需将该单元地址送给地址寄存器 J_{DN} 才能进行存取。地址包括二种情况，其一是指令地址，大家知道指令地址应由指令计数器 JS_Z 送往 J_{DN}。而数码地址则由变址加法器 Q_B 产生有效地址后再送往 J_{DN}。

2. 发送读写控制命令

通常、存贮器的主要操作只有两种，其一是存信息，其二是取信息，因此除发出地址以外，还需进行读写控制。通常，在控制器中设置有一个“读写控制触发器”C_{XN}，译码输出控制 C_{XN} 置“0”或置

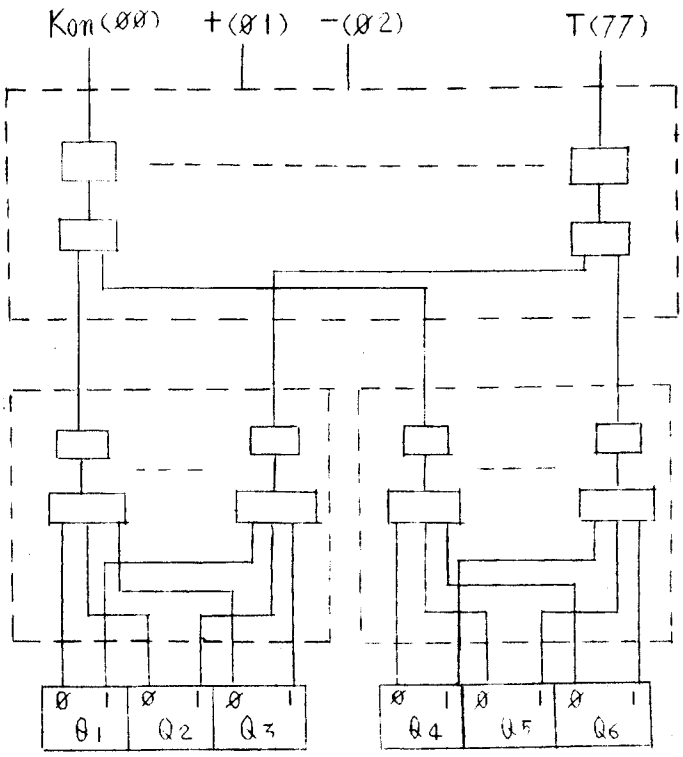

Fig 6－1

“1”。与此同时给内存一个启动信号m_{QN}，内存利用这个启动信号脉冲m_{QN}产生一系列的控制电位和脉冲，以控制其读写过程。

3. 发送写入代码

在存数的情况下，除上述以外，还需将欲存入之代码放到数码寄存器J_N，才能在发出写令后将此代码存入内存单元中。

4。“半字”触发器置“Ø”和置“1”

存取操作可以分为半字长操作和全字长操作，由于要使传送正确，故必须指明是半字长操作还是全字长操作，这由置“1”或置“0”